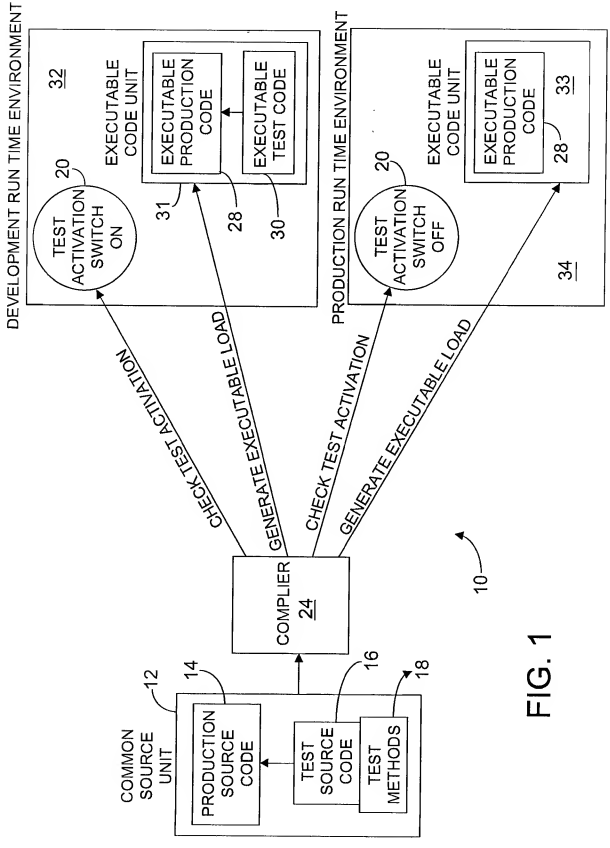
DEVELOPMENT RUN TIME ENVIRONMENT
32
EXECUTABLE CODE UNIT
EXECUTABLE PRODUCTION CODE
EXECUTABLE TEST CODE
31
28
30
TEST ACTIVATION SWITCH ON
20
PRODUCTION RUN TIME ENVIRONMENT
EXECUTABLE CODE UNIT
EXECUTABLE PRODUCTION CODE
33
28
TEST ACTIVATION SWITCH OFF
20
34
CHECK TEST ACTIVATION
GENERATE EXECUTABLE LOAD
CHECK TEST ACTIVATION
GENERATE EXECUTABLE LOAD
COMPLIER
24
10
COMMON SOURCE UNIT
12
PRODUCTION SOURCE CODE
14
TEST SOURCE CODE
16
TEST METHODS
18

FIG. 1

```
14a  * 1. productive class :
     * defination
     class OPERATIONS definition.
       public section.
         class-methods:
           ADD importing A type I
                         B type I
14         returning VALUE (RESULT) type I.
     endclass.

14b  * implementation
     class OPERATIONS implementation.
       method ADD.
         RESULT = A + B.
       endmethod.
     endclass.

18a  * 2. test class:
     * definition
     class TEST_OPERATIONS definition for testing.
       public section.
         methods TEST_ADD for testing.
     endclass.

     * implementation
     class TEST_OPERATIONS implementation.

       method TEST_ADD.

16   *    test data: variable needed to store the result from the productive method:
          data: ACTUAL_RESULT type I.

18b  *    call the method under test:
          ACTUAL_RESULT = OPERATIONS=>ADD ( A = 3  B = 5 ).

     *    compare the result with the expected value:
          CL_AUNIT_ASSERT=>ASSERT_EQUALS (
            ACT = ACTUAL_RESULT
            EXP = 8
            MSG = 'this is the message which occurs if the test failed'
          ) .

       endmethod.
     endclass.
```

FIG. 2

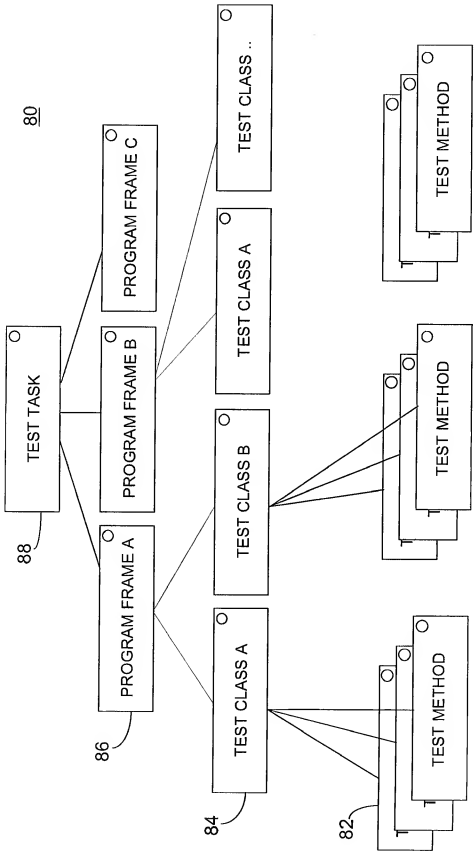
80
TEST TASK
88
PROGRAM FRAME A
PROGRAM FRAME B
PROGRAM FRAME C
86
TEST CLASS A
TEST CLASS B
TEST CLASS ..
84
TEST METHOD
TEST METHOD
TEST METHOD
82

FIG. 3

```
                56
ASSERT_EQUALS ( ACT = ACTUAL RESULT

                EXP = EXPECTED_RESULT
                MSG = 'this test has failed'
                QUIT = QUIT_VALUE ).
                57          58
```

Where QUIT_VALUE defines at which level the test flow should be interupted:

- NO: continue the current test method.
- METHOD: interrupt the current test method.
- CLASS: interrupt the test class execution.
- PROGRAM: abandon all test class executions of the currently tested program frame.

FIG. 4

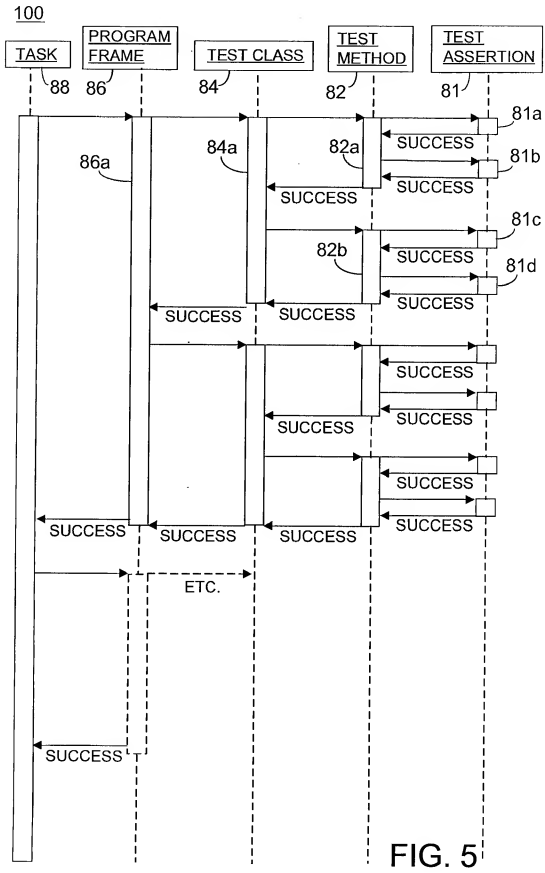
100
TASK
PROGRAM FRAME
TEST CLASS
TEST METHOD
TEST ASSERTION
88
86
84
82
81
81a
81b
81c
81d
86a
84a
82a
82b
SUCCESS
ETC.

FIG. 5

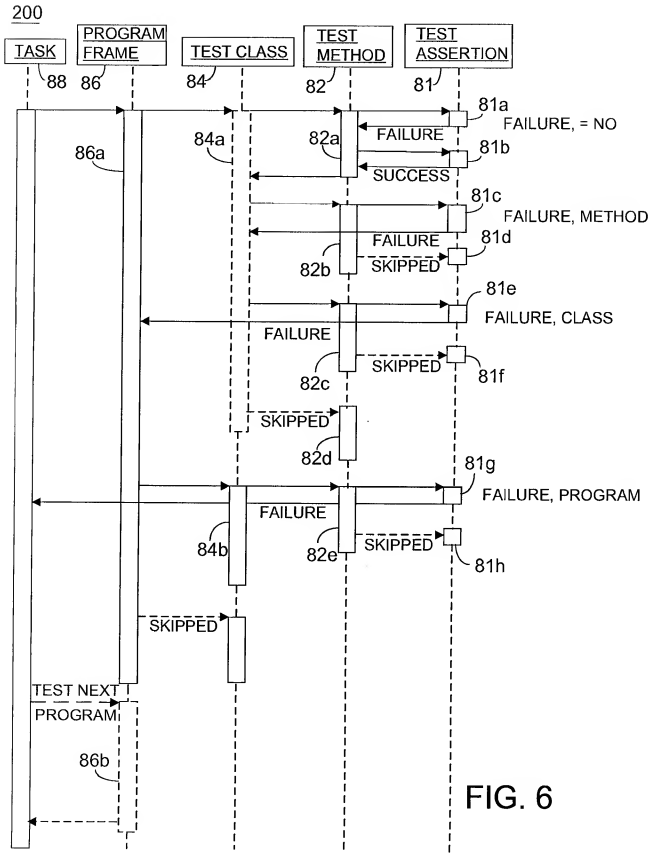
200
TASK
88
PROGRAM FRAME
86
TEST CLASS
84
TEST METHOD
82
TEST ASSERTION
81
81a
FAILURE, = NO
FAILURE
81b
SUCCESS
81c
FAILURE, METHOD
FAILURE
81d
SKIPPED
81e
FAILURE, CLASS
FAILURE
SKIPPED
81f
SKIPPED
81g
FAILURE, PROGRAM
FAILURE
SKIPPED
81h
SKIPPED
TEST NEXT PROGRAM
86a
84a
82a
82b
82c
82d
84b
82e
86b

FIG. 6